CLAIMS

What is claimed is:

1. A method for designing a software system in which system at least a first object is created arbitrarily earlier than a second object and said second object is automatically connected to at least said first object, said method comprising the steps of:

creating said first object;

creating a first container object capable of holding at least one other object of arbitrary object class;

defining at least a first template connection between said first object and said first container object;

creating said second object;

connecting said second object to said first object using said first template connection in which template said first container object is replaced with said second object.

2. The method in claim 1 wherein the step of creating said second object is performed by said first container object.

3. The method in claim 1 wherein the step of connecting said second object to said first object is performed by said first container object.

4. The method in claim 1 wherein the step of creating said second object is performed by said first container object and the step of connecting said second object to said first object is performed by said first container object.

5. The method in claim 1 wherein connections between all objects are established between connection points on said objects.

6. The method in claim 1 wherein said first template connection is defined in a data structure.

7. The method in claim 5 wherein said first template connection is defined in a data structure.

8. A system created using any one of claims 1, 2, 3, 4, 5, 6 or 7.

9. A method for describing connections between a plurality of objects in a software system in which at least a first object of said plurality is created

arbitrarily later than the remainder of said plurality, said method comprising the steps of:

defining at least a second object of said remainder;

defining a first container object which will be used as a placeholder for defining connections between said first object and said remainder;

defining at least a first connection between said second object and said first object by using said first container object in place of said first object.

10. A method for describing connections between a first plurality of objects in a software system and a second plurality of objects in said software system, said second plurality being created arbitrarily later than said first plurality, said method comprising the steps of:

defining at least a first object of said first plurality;

defining a first container object which will be used as a placeholder for defining connections between said first object and each object of said second plurality;

defining at least a first connection to be created between said first object and each object of said second plurality as a connection between said first object and said first container object.

11. In a software system, said software system having a plurality of objects, a container object comprising:

a first memory for keeping reference to at least a first object of arbitrary object class;

a section of program code causing said first memory to be modified so that it will contain a first reference to a second object;

a section of program code accessing a data structure and determining that at least a first connection needs to be established between said second object and at least a third object;

a section of program code causing said first connection to be established.

12. The container object of claim 10 further comprising a section of program code causing said second object to be created.

13. In a software system, said software system having a plurality of objects, a container object comprising:

a memory for keeping at least one reference to a contained object of arbitrary class;

a connection point for receiving requests to modify the set of contained objects;

at least one virtual connection point that accepts at least a first connection to be established to said contained object, said acceptance occurring before said contained object is added to said contained object;

a section of program code that establishes said first connection when said contained object is added to said container object.

14. In a software system, said software system having a plurality of objects, a container object comprising:

a first memory for keeping at least one reference to a contained object of arbitrary class;

a connection point for receiving requests to modify the set of contained objects;

at least one virtual property that accepts the value to be set in a first property on said contained object, said virtual property being capable of accepting values of a plurality of data types;

a section of program code that sets said first property on said contained object to said accepted value when said contained object is added to said contained object.

15. In a software system, said software system having a plurality of objects, a container object comprising:

a first memory for keeping a first plurality of contained objects of arbitrary classes;

a second memory for keeping a second plurality of unique identifiers, each identifier of said second plurality associated with exactly one object of said first plurality;

at least a first property, said first property being a second property of a first object of said first plurality and said first property being identified by a

combined identifier produced by combining the associated identifier of said first object and the identifier of said second property.

16. The software system of claim 15, wherein each said property comprises a terminal.

17. The software system of either of claims 15 or 16, wherein the second memory doesn't exist and contained objects are identified by identifiers assigned by the container.

18. A container object class in a software system, said software system having a first plurality of objects, each object of said first plurality belonging to an object class, said container object class comprising:

means for holding a second plurality of contained objects, said means being applicable to contained objects of any class;

means for changing the set of said contained objects, said means being applicable to contained objects of any class;

means for presenting said plurality of contained objects as a single object, said means being applicable to contained objects of any class.

19. The container object class of claim 18 wherein said single object is an instance of said container object class.

20. A container object which is an instance of the container object class of claim 18.

21. In a software system, said software system having a plurality of objects, each object of said plurality of objects belonging to an object class, said software system having means for building at least one structure of connected objects and means of describing said structure of connected objects, a container object class comprising:

means for holding a plurality of contained objects, said means being applicable to contained objects of any class;

means for changing the set of said contained objects programmatically, said means being applicable to contained objects of any class;

means for presenting said plurality of contained objects as a single object in said structure of connected objects, said means being applicable to contained objects of any class.

22. A container object which is an instance of the container object class of claim 21.

23. In a software system having at least a first object and a second object, said first object having at least one first connection point, said second object having at least one second connection point, said first connection point being used to establish a first connection between said first connection point of said first object and said second connection point of said second object, and said software system having means of requesting the establishment of a connection between connection points, a container object comprising:

means for adding and removing said first object from said container;

means for defining a third connection point on said container object;

means for transforming a requests for establishing of a connection between said second connection point and said third connection point into a request for establishing a connection between said second connection point and said first connection point.

24. The container object of claim 23 wherein said software system includes means of identifying said first connection point using a first identifier, said container object having the additional means to identify said third connection point using said first identifier.

25. The container object of claim 23 wherein said software system includes means of identifying said first connection point using a first identifier, said container object having the additional means to identify said first object using a second identifier and said container object having the additional means to identify said third connection point using a combination of said first identifier and said second identifier.

26. A container object in a software system, said software system having at least one first object and said container object, said first object having at least one first property, said software system having means of requesting operations over said first property, said container comprising:

means for adding and removing said first object from said container;

means for defining a second property on said container object;

means for transforming a request for operations over said second property into a request for operations over said first property.

27. The container object of claim 26 wherein said software system has means of identifying said first property using a first identifier, said container object having the additional means to identify said second property using said first identifier.

28. The container object of claim 26 wherein said software system has means of identifying said first property using a first identifier, said container object having the additional means to identify said first object using a second identifier and said container object having the additional means to identify said second property using a combination of said first identifier and said second identifier.

29. A container object having the sum of the means of the container object of claim 25 and of the container object of claim 28.

30. The container of claim 29 wherein all the specified means of said container are implemented independently of the class of said first object.

31. A container object in a software system, said software system having a plurality of objects, said software system having means for requesting operations over an object, said container object comprising:

means for holding a plurality of contained objects;

means for changing the set of said contained objects programmatically;

means for identifying each object of said contained objects by a separate, unique identifier for each object;

means of handling requests for operations over any object of said contained objects wherein said identifier is used to determine which object of said contained objects should handle the request.

32. The container of claim 31 wherein said container has the additional means of automatically assigning said unique identifier to each object added to said container.

33. The container of claim 31 wherein said unique identifier is assigned outside of said container, and said container has the additional means of associating said unique identifier with each said contained object.

34. A method for caching and propagating property values to a dynamic set of objects in a software system, said software system having a plurality of objects,

each of said objects having a plurality of properties, each said property having a value and an identifier, said method comprising the steps of:

accepting a first request to modify the value of a first property on behalf of said dynamic set of objects as if said dynamic set of objects were one object;

storing said value and identifier of said first property in a first data storage;

retrieving said value and identifier of said first property from said first data storage;

issuing a request to modify the value of said first property on a first object of said dynamic set of objects, using said value and identifier retrieved from said first data storage.

35. A container object in a software system using the method in claim 34.

36. A method for caching and propagating outgoing connections of a dynamic set of objects in a software system, said software system having a plurality of objects, said software system having means for establishing connections between said objects, said connections providing means for a first connected object to make outgoing calls to a second connected object, said method comprising the steps of:

accepting the request to establish a first outgoing connection between said dynamic set of objects and a first object, as if said dynamic set of objects were a single object;

storing a first data value necessary to effect said first connection in a first data storage;

retrieving said first data value from said first data storage;

issuing a request to establish a second connection between a second object of said dynamic set and said first object, using said first data value retrieved from said first data storage.

37. A container object in a software system using the method in claim 36.

38. A container object in a software system using both the method in claim 34 and the method in claim 36.

39. A container object in a software system, said software system having a plurality of objects, said software system having means for building at least one structure of connected objects, said software system having a first means of describing said structure, said container object being a first object in said structure, said first object having a first connection to at least a second object in said structure, said first connection being described by said first means, said container comprising:

means for holding a plurality of contained objects;

means for changing the set of said contained objects programmatically;

means for connecting each of said contained objects to said second object.

40. The container in claim 39 wherein said container has the additional means of establishing all connections between said container and other objects in said structure, said all connections being described by said first means, said additional means causing the establishing of each of said all connections between each of said contained objects and said other objects in said structure.

41. A container object in a software system, said software system having a plurality of objects, said software system having means of building at least one structure of connected objects, said software system having a first means of describing said structure, said software system providing a second means of enumerating all connections described by said first means, said container being a first object in said structure, said container being connected to at least a second object in said structure, said container comprising:

means for holding a plurality of contained objects;

means for changing the set of said contained objects programmatically;

means for finding a first described connection between said container and said second object;

means for establishing said first connection between a third object contained in said container and said second object.

42. The container in claim 41 wherein said container establishes connections between a first connection point of said third object and a second connection point of said second object.

43. A container object in a software system, said software system having a plurality of objects, said container having a first connection to at least one object, said first connection being described in a first data structure, said container comprising:

means for holding a plurality of contained objects;

means for changing the set of said contained objects programmatically;

means for determining a first set of connections to be established for each object added to said set of contained objects based on the set of connections described in said first data structure;

means for establishing said first set of connections.

44. The container in claim 43 wherein said container further comprises means for dissolving said first set of connections.

45. The container in claim 43 wherein said container further comprises:

means for remembering a second set of outgoing connections from said container to other objects

means for excluding said second set of connections from said first set of connections

means for establishing said second set of outgoing connections for each object added to said set of contained objects.

46. The container in claim 43 wherein said container further comprises:

means for remembering properties set on said container;

means for setting remembered properties on each new object added to said set of contained objects;

means for propagating properties set on said container to all objects in said set of contained objects;

47. A container object in a software system, said software system containing a plurality of objects, said software system having a first means to establish connections between connection points of objects of said plurality, said first means providing the ability to establish more than one connection to a first connection point of a first object, said container object having a second connection point connected to said first connection point of said first object, said container comprising:

means for holding a plurality of contained objects;

means for changing the set of said contained objects programmatically;

means for establishing a separate connection between a connection point on each object of said plurality of contained objects and said first connection point of said first object.

48. The container in claim 43 wherein said container further comprises:

means for remembering properties set on said container;

49. A part for distributing events among a plurality of parts, said part comprising:

a multiple cardinality input,

a multiple cardinality output,

means for recording references to parts that are connected to said output

means for forwarding events received on said input to each of the connected objects to said output.

50. A part for distributing events and requests between a plurality of other parts, said part comprising:

a first terminal for receiving calls;

a second terminal for sending calls out to a first connected part;

a third terminal for sending calls out to a second connected part;

means for choosing whether to send the received call through said second terminal or through said third terminal.

51. A part for distributing events and requests between a plurality of other parts, said part comprising:

a first terminal for receiving calls;

a second terminal for sending calls out to a first connected part;

a third terminal for sending calls out to a second connected part;

means for choosing whether to first send the received call through said second terminal and then through said third terminal or to first send the received call through said third terminal and then through said second terminal.

52. A part for distributing events and requests between a plurality of other parts, said part comprising:

a first terminal for receiving calls;

a second terminal for sending calls out to a first connected part;

a third terminal for sending calls out to a second connected part;

means for sending a first received call as a first call to said second terminal and then, based on value returned from said first call, choose whether or not to send said first received call as a second call to said third terminal.

53. A method for desynchronizing events and requests in a software system, said method comprising the steps of:

storing said event in a memory;

receiving a pulse signal;

retrieving said event from said memory and continuing to process said event in the execution context of said pulse signal.

54. A part in a software system, said part comprising:

a first terminal for receiving calls;

a second terminal for sending calls out to a first connected part;

a third terminal for receiving a pulse call;

a memory for storing call information received from said first terminal;

a section of program code that is executed when said part receives said pulse calls, said section retrieving said call information from said memory and sending a call out to said second terminal.

55. The part in claim 54 wherein said memory can hold call information for a plurality of calls.

56. The part in claim 54 wherein said memory is a queue.

57. The part in claim 54 wherein said memory is a stack.

58. The part in claim 54 wherein said first terminal and said second terminal are one terminal.

59. A part in a software system, said part comprising:

a first terminal for receiving calls;

a second terminal for sending calls out to first connected part;

a memory for storing call information received from said first terminal;

a means for obtaining execution context;

a section of program code that is executed in said execution context, said section retrieving said call information from said memory and sending a call out to said second terminal.

60. The part in claim 59 wherein said means for obtaining execution context is a thread of execution in a multithreaded system.

61. The part in claim 59 wherein said means for obtaining execution context is a timer callback.

62. The part in claim 59 wherein said means for obtaining execution context is a subordinate part.

63. The part in claim 59 wherein said means for obtaining execution context is a subordinate part, said subordinate part having a primary function of providing execution context for other parts.

64. The part in claim 59 wherein said first terminal and said second terminal are one terminal.

65. A part in a software system, said part comprising:

a first subordinate part for storing incoming data;

a second subordinate part for generating execution context.

66. The part in claim 65 wherein said part further comprises a connection between said first subordinate part and said second subordinate part.

67. A part in a software system, said part comprising:

a first terminal for receiving an incoming request;

a second terminal for sending out an outgoing request;

a third terminal for receiving a request completion indication;

a synchronization object for blocking the thread in which said incoming request was received until said request completion indication is received.

68. The part in claim 67 wherein said second terminal and said third terminal are one terminal.

69. A part in a software system, said part comprising:

an input terminal for receiving calls of a first type;

an output terminal for sending calls of a second type;

means for converting calls of said first type to calls of said second type.

70. A part in a software system, said part comprising:

an input terminal for receiving calls of a first type and sending calls of said first type;

an output terminal for receiving calls of a second type and sending calls of said second type;

means for converting calls of said first type to calls of said second type;

means for converting calls of said second type to calls of said first type.

71. The part of claim 70 wherein said first type and said second type differ by physical mechanism.

72. The part of claim 70 wherein said first type and said second type differ by logical contract.

73. A part in a software system, said part comprising:

a first terminal for receiving a first request and sending a second request;

a second terminal for sending said first request;

a third terminal for receiving said second request.

74. The part of claim 73 wherein:

said first terminal is a bidirectional terminal;

said second terminal is an output terminal;

said third terminal is an input terminal.

75. A part in a software system, said part comprising:

a first terminal for receiving calls;

a second terminal for sending out calls received on said first terminal;

a third terminal for sending out calls whenever a call is received on said first terminal.

76. The part in claim 76 wherein said part further comprises a first property for defining a criterion for selecting for which calls received on said first terminal said part will send out calls through said third terminal.

77. The part in claim 76 wherein said part further comprises a second property for configuring what call said part will send out said third terminal.

78. The part in claim 76 wherein said part further comprises a third property for configuring what call said part will send out said third terminal before sending out a call received on said first terminal to said second terminal.

79. The part in claim 76 wherein said part further comprises a third property for configuring what call said part will send out said third terminal after sending out a call received on said first terminal to said second terminal.

80. The part in claim 76 wherein said part further comprises a third property for configuring whether a call out through said third terminal should be made before or after sending out a call received on said first terminal to said second terminal.

81. A part in a software system, said part comprising:

a first terminal for receiving calls;

a second terminal for sending out calls received on said first terminal;

a third terminal for sending out calls whenever a call sent out said second terminal returns a pre-determined value.

82. The part of claim 81 wherein said part further comprises a property for configuring said pre-determined value.

83. The part of claim 81 wherein said pre-determined value indicates that said second call has failed.

84. The part of claim 81 wherein said pre-determined value indicates that said second call has succeeded.

85. A part in a software system, said part comprising:

a first terminal for receiving calls;

a second terminal for sending out calls received on said first terminal;

a first property for configuring a first value;

a third terminal for sending out notification calls whenever a call sent out said second terminal returns a second value that matches said first value.

86. The part of claim 85 wherein said part further comprises a second property for configuring whether said part will send out said notification calls if said second value matches said first value or if said second value differs from said first value.

87. A part in a software system, said part comprising:

a terminal for receiving calls of arbitrary logical contract;

a property for defining a return value.

88. The part of claim 87 wherein said part further comprises a property for configuring the logical contract for calls received on said terminal.

89. The part of claim 87 wherein said terminal is an input terminal.

90. The part of claim 87 wherein said terminal is a bi-directional terminal and said part does not make calls out said terminal.

91. A part in a software system, said part comprising:

a terminal for receiving a first call and a reference to a first memory;

a property for defining a return value;

a section of program code for freeing said first memory.

92. The part in claim 91 wherein said part further comprises means for determining whether said section of program code should be executed for said first call.

93. The part in claim 91 wherein said part further comprises means for determining whether said section of program code should be executed for said first call based on a value contained in said first memory.

94. A part in a software system, said part comprising:

a first terminal for receiving a first call;

a second terminal for sending out said first call;

means for extracting data from said first call;

means for formatting said extracted data as a first text;

means for sending out said first text.

95. The part of claim 94 wherein said means for sending out said first text is a third terminal.

96. The part of claim 94 wherein said means for sending out said first text is a section of program code that invokes a function for displaying said first text on a console.

97. A first structure of connected parts in a software system, said first structure comprising:

a factory part for determining when a new part should be created;

a container part for holding a first plurality of parts of arbitrary part class;

a connection between said factory part and said container part.

98. The structure of claim 97 wherein:

said factory part has a first terminal;

said container part has a second terminal;

said connection is established between said first terminal and said second terminal.

99. The structure of claim 97 wherein said structure further comprises a demultiplexing part having a first terminal for receiving calls, a second terminal for sending out calls and means for selecting a part connected to said second terminal.

100. The structure of claim 99 wherein said structure further comprises a plurality of connections, each connection established between said second terminal of said demultiplexing part and a terminal of each part in said first plurality.

101. The structure of claim 100 wherein said connection demultiplexing part and said factory part are one part.

102. A composite part in a software system, said composite part comprising the structure in claim 97.

103. The structure of claim 97 wherein said structure further comprises an enumerator part for defining the set of parts in said first plurality.

104. The structure of claim 103 wherein said structure further comprises a connection between said enumerator part and said factory part.

105. The structure of claim 97 wherein said enumerator uses a data container for defining the parts in first plurality.

106. The structure of claim 103 wherein said enumerator comprises means for enumerating a set of peripheral devices connected to a computer system.

107. The structure of claim 106 wherein said enumerator further comprises a first property for configuring a limitation on the type of peripheral devices to be enumerated.

108. The structure of claim 97 wherein said structure further comprises a parameterizer part for retrieving the value for at least one property to be set on each part of said first plurality.

109. The structure of claim 108 wherein said parameterizer part retrieves said value from a data container.

110. The structure of claim 108 wherein said parameterizer part uses a persistent identifier to select said value among a set of values.

111. The structure of claim 97 wherein said structure further comprises a serializer part for saving the value of at least on property of each part in said first plurality.

112. The structure of claim 111 wherein said structure further comprises a trigger part for initiating said saving of the value.

113. The structure of claim 97 wherein said structure further comprises a parameterizer part for retrieving the value for a first property to be set on each part of said first plurality and for saving the value of said first property.

114. The structure of claim 97 wherein said factory part determines whether to create a new part in said first plurality or to use an existing part in said first plurality based a persistent identifier provided to said factory part.

115. The structure of claim 97 wherein said structure further comprises a loader part for bringing in memory a class for a part to be created.

116. The structure of claim 116 wherein said structure further comprises:

a connection between said factory part and said loader part;

a connection between said loader part and said container part.

117. A part in a software system, said part comprising:

a first terminal for receiving calls;

a second terminal for sending out calls received on said first terminal;

a third terminal for sending out requests to create new parts;

means for selecting calls received on said first terminal for which said part sends out requests on said third terminal.

118. A method for designing access to a hardware component in a component-based software system, said method comprising the steps of:

designating a first software component for receiving interrupts from said hardware component;

designating a at least a second software component for accessing input and output ports of said hardware component;

designating a third software component for handling interrupts received by said first software component;

designating a fourth software component for manipulating said hardware component;

connecting said first software component to said third software component;

connecting said second software component to said fourth software component.

119. The method in claim 118 wherein said method further comprises the step of connecting said third software component and said fourth software component.

120. The method in claim 118 wherein said third software component and said fourth software component are one component.

121. A part in a software system, said part comprising:

a first terminal for sending out calls;

a section of program code for receiving control when an interrupt occurs and sending out a call through said first terminal.

122. The part of claim 121 wherein said part further comprises a property for configuring which hardware interrupt vector among a plurality of hardware interrupt vectors said part should receive.

123. The part of claim 121 wherein said part further comprises a section of program code for registering said part to receive control when said interrupt occurs.

124. A part in a software system, said part comprising:

a terminal for receiving requests to access at least one port of a hardware component;

a property defining the base address of said port;

a section of code that accesses said port when a request is received on said first terminal.

125. The part of claim 124 wherein said port is a memory-mapped port.

126. The part of claim 124 wherein said port is a input-output port.

127. The part of claim 124 wherein said requests include a read request and a write request.

128. A structure of connected parts in a software system, said structure comprising:

an interrupt source part for receiving interrupt from a hardware component;

at least one port accessor part for accessing ports of said hardware component;

at least one controller part for controlling said hardware component.

129. The structure of claim 128 wherein said controller part accesses said hardware component exclusively through said interrupt source part and said port accessor part.

130. The structure of claim 128 wherein said structure further comprises:

a connection between said interrupt source part and one of said controller parts;

a connection between one of said port accessor parts and one of said controller parts.

131. A composite part in a software system, said composite part containing the structure of claim 128.

132. A composite part in a software system, said composite part containing the structure of claim 129.

133. A method for designing software system in which system at least a first object is created arbitrarily earlier than a second object and said second object is automatically connected to at least said first object, said method comprising the steps of:

creating said first object;

creating a first container object capable of holding at least one other object of arbitrary object class;

defining at least a first template connection between said first object and said first container object;

creating said second object;

connecting said second object to said first object using said first template connection in which template said first container object is replaced with said second object.